

# A blueprint for business architectures

**IBM's SNA Distribution Services represent one path to information network architectures, which promise to free users from the constraints of time and location.**



**T**oday, service-industry applications consist of heterogeneous environments that arose because buyers favored "specialized" computer vendors. Engineering groups tended to purchase from Hewlett-Packard and Digital Equipment Corp., office processing groups from Wang, and accounting-oriented groups from IBM. Accordingly, each vendor developed protocols that worked only with its gear.

To overcome such isolation and allow users or programs from any one environment or on any piece of equipment to communicate with users in any other environment, information network architectures (INAs) will have to be established in five to 10 years. Certain users, the "Citibanks" of the world, have already begun to develop their own. IBM has tipped its INA hand with strategic directions in its office product line, based on its Systems Network Architecture (SNA). Other vendors are likely to follow the path being blazed by international standards bodies.

The purpose of an INA is to separate business application programs from the communications support structure and to provide these programs with a common interface to obtain end-to-end services. This necessitates dividing what is called the application layer in many existing network architectures into three distinct layers. Properly executed, an INA could allow users to share files and information, based not on which minicomputer or mainframe host (or file server) the users are connected to, but on a common area of interest or activity.

To understand the need for an INA, consider the predicament of many users. Business application programs were written for individual departments having different hosts (minicomputers or mainframes), each of which had its own unique communications software components. Application programs (the "A" programs

in Figure 1) assumed such communications functions as routing, protocol and device management, and polling. In some cases, the A program itself would perform the communications functions. In others, the communications subprogram of the A program, the "C" portion, would do so. Boundaries between A and C programs were not uniform across terminal types.

Whether located in the A or the C portion, one element in the host application controls and acts on behalf of the terminal. In order to access the application, the terminal must communicate with this element. Note that the logical connection is established in the host, where both the terminal's element and the application processing reside, although the physical connection is out in the real world. Therefore, there is a logical star "network" in which all logical connections terminate in the host.

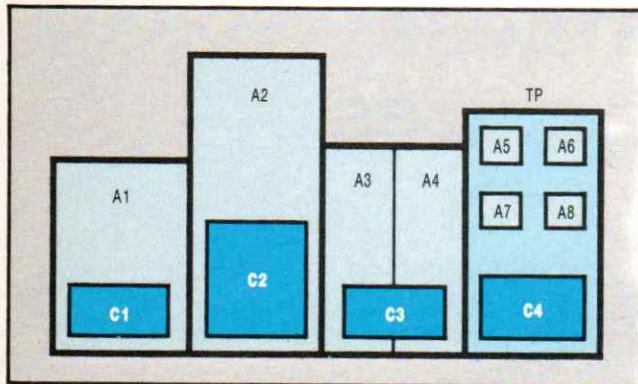
With this type of logical network, each application program controlled its own terminal resources, making it impossible to share communications programs or information across different applications or across different corporate departments. Even if a connection could be made, information content could not necessarily be exchanged intact. Each application had its own telephonelike network (see "Evolution of a fractured mish-mash"). For one terminal type, all the polling and message decomposition occurred within the A portion. For another, the C function might have done the polling, but the A itself did the message decomposition. Another problem could have been the use of different versions of the same data communications protocol.

The type of communications interface that evolved, that is, each of the legs of the logical network, was point-to-point. This approach corresponded with the corporate structure of independent business units re-



**1. Discombobulation.** In today's environment, the communications portions of application programs are dedicated and inconsistent from one to the other.

HOST (MINICOMPUTER OR MAINFRAME)



A = APPLICATION PROGRAM  
C = COMMUNICATIONS PROGRAM

TP = TELEPROCESSING MONITOR

sponsible for their own profit and loss as well as for computer resources. For example, large banks had different departments, such as credit cards, personal loans, and checking and saving accounts, operating completely independently from each other. In support of such applications and of data processing at large, companies developed point-to-point schemes.

In addition, two other areas fostered this development: office computers and microcomputers. Office computers and their associated word processing terminals were introduced into corporations primarily as flexible and highly productive "typewriters." They were not implemented with the expectation that they would later actively participate in corporate networks. However, the view is becoming more widely held that office computers can be used for correspondence among professionals.

Although this perception is creating a need for electronic mail and document interchange, broader acceptance of these services must await computers that function as post offices—distributing mail to other post office computers, which is then accessed by users through their terminal devices. Any company with two or more offices and computers in each will need to share information between the machines. A document entered at a terminal will likely be sent to another user, with the computers acting as intermediary.

The explosive proliferation of microcomputers has brought the problems associated with point-to-point arrangements into sharp focus. Previously, users depended totally on mainframes or minicomputers for business applications, even for word processing. With the advent of the microcomputer, they could access friendly, sophisticated, powerful applications of all types right at their desks.

Most users have become aware that, to realize the full potential of the microcomputer, they also need access to corporate databases. In the first year of using the microcomputer, for example, users might

## Evolution of a fractured mish-mash

The earliest application programs supporting terminal networks integrated the logic for business-oriented task processing (for which the application was initially designed) with the logic used for communications. There was no separate communications program. The communications program was embedded in the application program and ran as a subroutine, as depicted in the figure (part a). Lines and terminals could not be shared with other programs.

For example, BTAM (Basic Telecommunications Access Method), IBM's oldest communications program, required that lines and terminals be dedicated to the application served by the access method. Therefore, each application had to have its own copy of BTAM. Perhaps more importantly, users had to have a separate terminal on their desks for each application.

To alleviate this problem, communications software, such as teleprocessing (TP) monitors that permitted the sharing of communications resources, was developed. TP monitors were large programs containing smaller ones (see figure, part b). Customer Information Control System (CICS) is one example of a TP monitor.

This intermediate implementation provided considerable relief because the smaller programs could share the communications resources that were now owned on their behalf by the TP monitor. However, while programs running under a common TP monitor could now share lines and terminals, no sharing was possible between different TP monitors.

The next step was to have a separate program for communications with business programs connected to it (see figure, part c). An early IBM example is the Telecommunications Access Method (TCAM). Whenever the application program needed to communicate with terminals, it asked the communications program to provide the appropriate resources.

For user organizations with monolithic, self-contained needs, this approach sufficed for a while. However, even such organizations ran into limitations as they developed a mix of TP monitors and other types of programs. The need to share terminals between multiple applications in the same computer or with applications in another computer stimulated intense development activities by manufacturers. These efforts resulted in a complete overhaul of the means by which information is exchanged between terminals and application programs.

Communications architectures are an outgrowth of this era. They were developed to segment communications functions and to standardize interfaces. These functions were broken down into layers, of which the lower ones were responsible for the actual movement of data and the upper ones for the establishment of connections, or sessions, between terminals and programs. Each layer has a correspondent, or peer, layer in the other node, with which it believes it is in direct dialog. The actual flow, however, goes



through the layers and back again.

Since each vendor developed its own architecture, different implementations that were incompatible with one another became available. To ensure compatibility, vendors realized that they would have to keep up with the enormous resources of the dominant vendor (IBM) and with continuous enhancements to IBM's Systems Network Architecture (SNA). Such knowledge is both rare and expensive. Given the exorbitant costs faced by non-IBM vendors, most have begun to adopt the architecture sponsored by the International Organization for Standardization (ISO), namely, the Open Systems Interconnection (OSI) reference model. It is likely that, by the end of the decade, implementations of IBM's SNA and ISO's OSI will have come to dominate the marketplace.

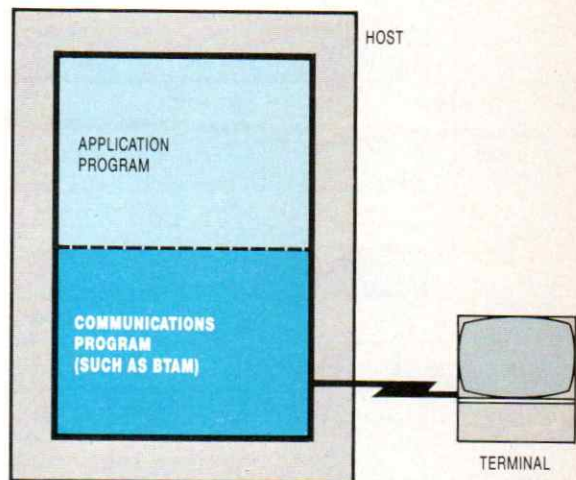
The most important consideration for users is that both architectures, or at least interfaces into them, will be running on their computers. SNA will be supplied by the dominant vendor; OSI implementations will represent the rest of the world. But such implementations may be available even from IBM in the future. IBM recently announced a new product called "Open Systems Transport and Session Support," which complies with OSI layers four and five. The company is currently evaluating the top two layers of the OSI model: six (presentation) and seven (application). As the specifications of these layers are completed, IBM may find it advantageous to develop products supporting those layers.

A major dividend of the development of formal architectures was the interconnection of multiple hosts. It thus became necessary for the programs responsible for terminals on one host to access application programs of other hosts. Routing and resource scheduling between computers became a major issue. IBM initially accomplished interconnection within a single network using Multiple Systems Networking Facility (MSNF) software and later, across independent networks, using the SNA Interconnect (SNI) package.

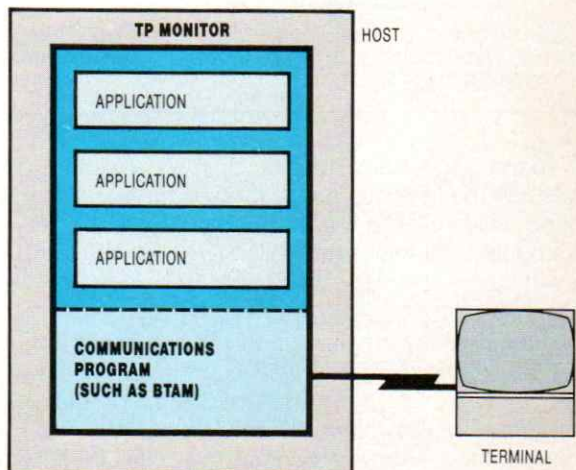
Since its announcement, IBM's SNA (today the most successful network architecture in the marketplace) took 10 years to become workable, reliable, and acceptable. Yet, with the notable exception of advanced program-to-program communications (APPC), SNA has only solved the problem of communications that are sent from a terminal to programs in one or more host computers.

With APPC, IBM is finally beginning to respond to the proliferation of intelligent nodes in the network. (Such nodes include microcomputers, distributed office computers like the System/36 and System/38, and even IBM local area networks, which can be considered distributed computers.) APPC is a protocol under SNA that uses logical unit (LU) 6.2 sessions to enable a program to exchange information with another program rather than with a terminal.

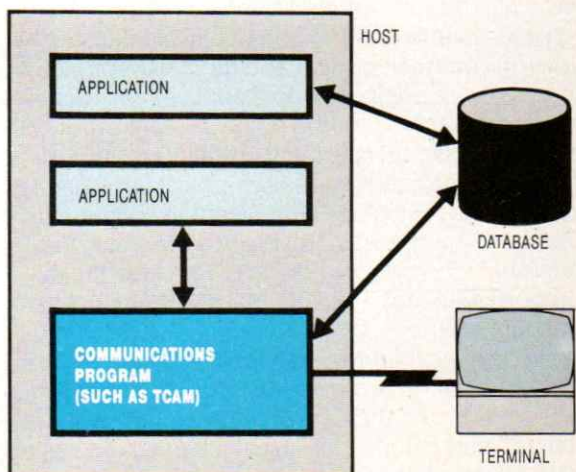
(A) DEDICATED COMMUNICATIONS RESOURCES



(B) TELEPROCESSING (TP) MONITORS



(C) STANDALONE COMMUNICATIONS PROGRAMS

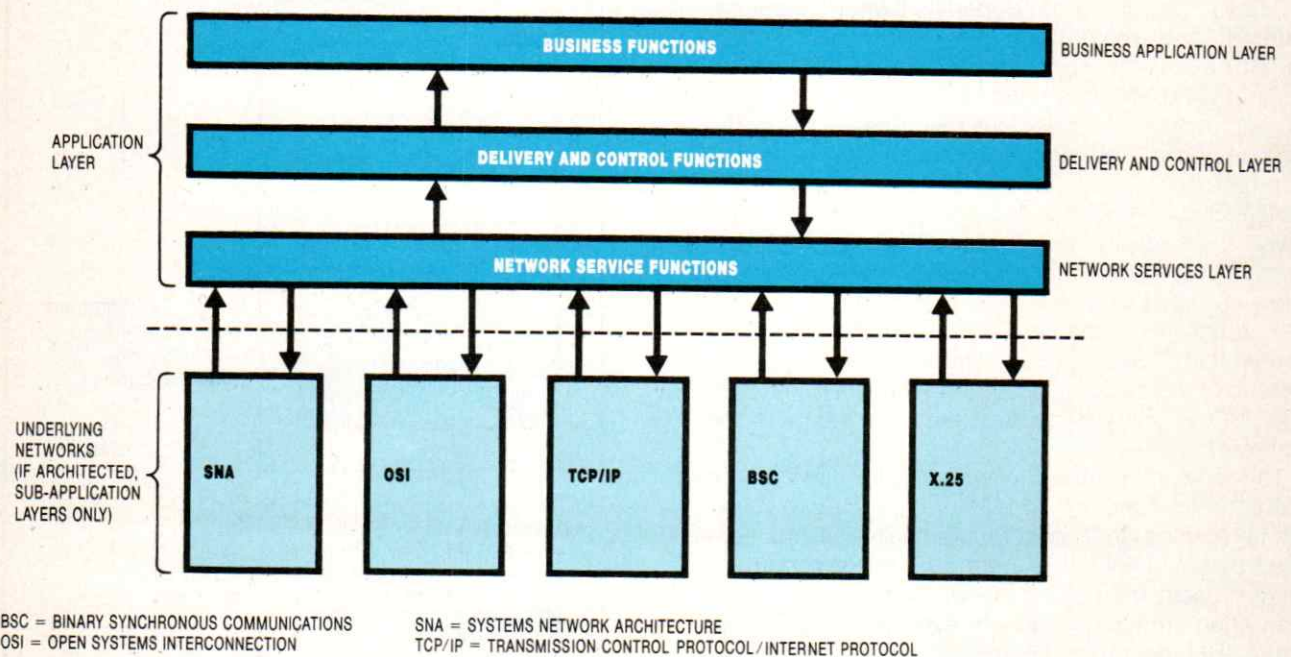


BTAM = BASIC TELECOMMUNICATIONS ACCESS METHOD  
TCAM = TELECOMMUNICATIONS ACCESS METHOD



**2. Architectural solution.** Dividing the application layer into three parts gives independence from lower layers and better interprogram communications. The amount

of code in the lowest of the three new layers, which rest on top of existing architectures, varies with the maturity of the underlying layer.



wish to massage mainframe-resident data with Lotus 1-2-3. But in the second year, they look for an alternative to hand delivering the massaged data. Only then do users begin to demand the ability to exchange information with each other and with computers other than the attached mainframe.

Almost all of the smaller machines had some communications capability but only with the IBM host and mostly by emulating nonintelligent 3270 terminals. Manufacturers and software houses rushed to provide quick, patched, short-sighted, and yet relevant solutions to some of the information-exchange problems faced by the user community. As a result, many point-to-point implementations linking a certain microcomputer to a certain mainframe or one communicating word processor to another became available.

However, with the growth and rapid acceptance of various types of computers and applications as well as rising user expectations, the point-to-point products created the classical spaghetti network of short, dead-end roads. Such roads were both physical, linking two machines, and logical, as between host software components. No access was provided to the electronic "interstate highways" of X.25 and T1, since these backbones were not well integrated with the point-to-point spaghetti.

Networks were highly specialized, separate entities. Terminals clustered around a mainframe or minicomputer that acted as the network host. Alternately, self-contained, standalone local area networks (LANs) connected different hardware via a common medium, such as fiber, twisted pair, or coaxial cable, with no information sharing across networks. Communications in these LANs, including physical interface, data link control,

and physical connection services, have been essentially standardized. However, the problems of information sharing, including the structure of content (such as common file and document formats) and internetwork delivery, are not yet resolved. Moreover, each network is managed separately.

### The way out

To help corporations operate their businesses in the future, networks will have to allow for the free transfer of information, regardless of manufacturer and product type. From the user's perspective, the next step in computer evolution that needs to occur should be the development of a common, networkwide function known as "information delivery." This refers to the interconnection of applications, regardless of their location or internal format or the type of computer on which they reside.

Figure 2 shows the layers of INA that users will need to secure for themselves in the coming decade. (IBM's SNA Distribution Services, or SNADS, will become a de facto INA standard. Users too small to develop their own INAs should request them from their vendors.) The separation of the application layer into three layers each having a discrete function can be seen as an extension of current architectural specifications. It would provide a common "pipeline" supporting services for storing, routing, protecting, and accounting for information between application programs, roughly comparable to the services of the U. S. Post Office for delivering mail.

Typically, standard architectural reference models depict the application layer as one single entity. This view resulted from the fact that, in the past, each



application program had a unique communications interface—as discussed above.

IBM has begun to build a mechanism for information delivery based on SNADS (see “SNADS: The beginning of IBM’s INA?”). While such delivery today includes only text documents, there is little doubt that it will be extended to, for example, voice, graphics, and imagery. An important question is whether or not other computer companies will also adopt the same technique in a compatible manner to solve the same problem. As IBM is evolving its delivery infrastructure, other standards—such as X.400 from the International Telegraph and Telephone Consultative Committee (CCITT)—are being accepted by some companies. This may only add to the confusion surrounding SNA versus Open Systems Interconnection (OSI).

User organizations will find it extremely difficult to begin converting millions of dollars worth of software to utilize SNADS with LU (Logical Unit) 6.2 or X.400 with OSI today. However, to wait and see which will be the predominant way to implement a delivery mechanism will only increase the risk and cost. Choosing one now makes better sense. Users could either develop their own delivery infrastructure, if they have the resources (Citibank, for example, has already begun to do this), or locate one developed and supported by an independent third party who could support the multiple vendors (or equipment types) in the user’s installation.

### Details of the architecture

INA breaks down the functions of the application layer and redefines their relationship with presentation layer functions for the purpose of information sharing. Today, the relationship between the application and presentation layers is not uniform because the boundaries and functions between the two layers have not been clearly defined. Different vendors and standards bodies are working together, through the International Organization for Standardization (ISO), to define the upper layers of the OSI architecture.

Since OSI is a reference model, each vendor implementation of it is likely to be different. Even when the reference model is fully defined, implementations may be different or inconsistent because programmatic interfaces between applications and the presentation layer are different. And even if these interfaces were the same, both within and across vendors, there is yet another level of the application functions to consider: the decomposition and mapping of transactions into atomic business functions.

For example, airline reservations consist of several atomic transactions. An incoming ticket request is decomposed into three transactions: one to update the seat inventory on the flight, one for the financial exchange, and one for issuing the ticket. Each of these atomic transactions is individually processed by its respective business program. Each atomic business program, in turn, responds to the code responsible for orchestrating the process. The orchestrator initiates each atomic business program as it is required. When all atomic transactions have been processed, the orchestrator responds to an airline terminal with the result

of the request (for example, no seats, credit denial, printer broken, request completed). However, this division of labor among discrete routines might exist only in a limited number of application programs today. Where it does exist, no two applications use the same structure.

Users within both single- and multivendor environments must protect themselves from this lack of uniformity. One way of doing so is by breaking down and carefully defining the application layer into three layers known as business application, delivery and control (D&C), and network services.

### What’s at the layers

The top layer is responsible for the business functions of application programs. For example, one business function would be the reading and writing of files (credit inquiries, account balances) to and from disk. It should not have to be concerned about where the files or the target disk reside in the network. On behalf of business functions, the D&C layer requires access to a directory that knows the name and location of the appropriate computer, whether local or remote. Application programs can address local and remote resources automatically.

Business functions use the services provided by the D&C functions. One D&C function might be to package a file and ship it to its destination. Services provided by the delivery layer will allow business functions to communicate on a peer-to-peer basis with other business functions. For example, a transfer of funds in one business function may trigger a debit in another in the same computer or on a distant computer. This process requires proper routing decisions by the delivery layer. Moreover, different business functions may require different classes of services, which corporations should be able to assign.

The last layer, network services, is responsible for functions typically thought of as communications. It must provide interfaces to architected protocols, such as Systems Network Architecture (SNA), Open Systems Interconnection (OSI), and Transport Control Protocol/Internet Protocol (TCP/IP), as well as nonarchitected protocols, such as binary synchronous communications (BSC). The network services layer packages messages into the appropriate format and attaches whatever routing information (headers and so forth) is required.

The primary purpose of this layer is to insulate investments in code written for business functions and D&C functions from the computer hardware environment as well as from public and proprietary data communications protocols. This allows business functions to be redesigned or rewritten without affecting the underlying transport mechanism. Moreover, it lets corporations integrate business functions throughout multiple interconnected networks, regardless of the protocol used by each network.

The amount of code in the network services layer is related to the lack of uniformity of the presentation-layer implementations available today. As presentation layers become more uniform, the need for the network



## SNADS: The beginning of IBM's INA?

To date, Systems Network Architecture (SNA) has mostly addressed problems particular to terminals sharing application programs in one or more mainframes. However, IBM has begun to respond to the market's demand for new ways to use its gear and software for information networking. As part of this strategy, IBM has created and is starting to implement enhancements corresponding to delivery and control functions under the name of SNA Distribution Services (SNADS). As such, SNADS may represent an IBM bid to supply an information network architecture (INA) to businesses.

IBM announced SNADS for store-and-forward document distribution between multiple mainframes running the Distributed Office Support System (DISOSS), IBM's host-based archive. SNADS, currently bundled with DISOSS, allows documents to be scheduled for distribution to another DISOSS host even when the path to that host is not active. In addition to its archival or library functions, DISOSS also serves as an intermediate storage location.

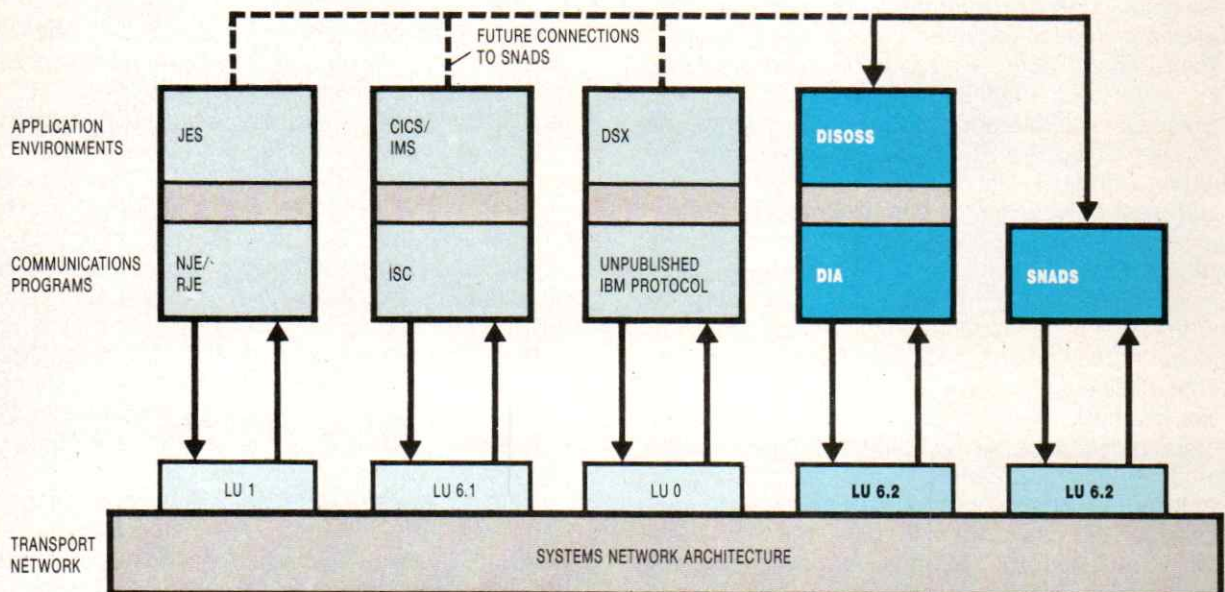
SNADS uses IBM's Document Interchange Architecture (DIA) and Document Content Architecture (DCA) for message handling. DIA is a specialized communications protocol for mailing objects. It specifies the rules for delivery and defines the envelope carrying the attributes both of the object and of its recipients. DCA specifies and defines the structure of documents (with such parameters as layout, indentation, and margin justification). Currently, this structure is quite simple. DIA and DCA can only deliver documents consisting of text. However, in the future, incoming data will contain various types of information objects, such as text, voice, graphics, and imag-

ery, forming a compound document.

DISOSS can be considered a local post office or an intermediate mailroom. When one DISOSS mainframe talks to another, it uses SNADS as an interpost-office protocol when the transaction is store-and-forward. (For immediate relaying of messages, standard SNA sessions are established.) Links between DISOSS machines through SNADS are set up using the Logical Unit (LU) 6.2. All user messages are packaged in DIA and submitted to the node that implements the local post office functions of DISOSS (a subset of DISOSS known as the internal messaging application). The messages may then either be distributed locally by that DISOSS host or they can be handed over to SNADS for delivery to another.

The figure shows that every IBM application with its associated devices presently requires its own unique communications program and SNA logical connection (represented by different LUs). For example, remote job entry (RJE) or network job entry (NJE) provide access for remote printers and workstations accessing the Job Entry Subsystem (JES), a batch-oriented application. At the transport level, JES and remote devices depend on LU type 1. Other applications and devices similarly rely on highly product-dependent LU implementations. Another disadvantage is that the LUs require logical point-to-point connections. These connections are an outgrowth of multiple delivery mechanisms, each of which is both application- and device-dependent.

For IBM to paint itself out of this corner, it needed to move toward another type of LU. This LU would let any program or user talk to any other without requiring a direct logical connection. Instead, it would let a



CICS = CUSTOMER INFORMATION CONTROL SYSTEM  
DIA = DOCUMENT INTERCHANGE ARCHITECTURE  
DISOSS = DISTRIBUTED OFFICE SUPPORT SYSTEM  
DSX = DISTRIBUTED SYSTEMS EXECUTIVE

IMS = INFORMATION MANAGEMENT SYSTEM  
ISC = INTER-SYSTEM COMMUNICATIONS  
JES = JOB ENTRY SUBSYSTEM  
LU = LOGICAL UNIT

NJE = NETWORK JOB ENTRY  
RJE = REMOTE JOB ENTRY  
SNADS = SNA DISTRIBUTION SERVICES  
(SNA = SYSTEMS NETWORK ARCHITECTURE)



user address a common piece of code in each computer. LU 6.2 is a step in this direction.

However, the way LU 6.2 is implemented today still requires that every node in the network implement a logical session manager. For LU 6.2 to work, this implies that a virtual point-to-point connection must still be set up between two nodes. Currently, advanced program-to-program communication (APPC) does not address the higher-level functions needed to form a real information network, such as alternate routing, time-independent delivery, and the mapping of canonical forms. Thus program-to-program communications and LU 6.2 do not as yet solve the problem of information delivery. In order to reap the true benefits of LU 6.2, IBM will have to modify the logical unit so that it is decoupled from its current environment and transformed into a common resource that can be addressed by entities throughout the network. This resource will provide a set of network services in support of information delivery and control.

SNADS, which is evolving into a delivery and control layer, will utilize this networkwide code in order to locate programs, establish and control the connections, and deliver information. SNADS also needs to be decoupled (from DISOSS) because each node does not need a full version of DISOSS but merely its routing portion.

By means of SNADS, a user at a microcomputer that is not configured as an RJE device will eventually be able to submit a request to JES, even if the microcomputer is not directly connected to the mainframe running JES. A batch user could send print requests to any printer in the network regardless of whether that printer supports a JES device (such as 2770, 3770, 2780, or 3780) and regardless of whether the necessary resources are active at the time. SNADS will also permit network operators to dynamically configure new printers without having to reconfigure JES, as is required with RJE and LU 1.

SNADS is one of the first industry implementations of delivery and control functions. Since IBM has opened up SNADS to the non-IBM world (that is, published the SNADS formats and protocols so that non-IBM vendors can implement compatible functions), SNADS has a good chance to become a de facto standard just like SNA. But, typically, objects must be moved between more than just an IBM and a non-IBM environment. The real question is, who will deliver objects from DEC to Wang?

The authors believe that non-IBM vendors that are promoting their processors at the departmental level will need to implement DISOSS-like archival and SNADS-like distribution services. It is unlikely that these companies will develop their own proprietary versions of such a delivery mechanism that do not interface with the evolving IBM delivery and control layer. Such software is currently being developed by the non-IBM vendors, often in conjunction with third-party software houses.

services layer will be reduced. The extent of the layer is also related to the sophistication of the underlying protocol. Today, for example, the amount of code within the network services layer for BSC would be greater than that for SNA to compensate for BSC's lack of a presentation layer.

### **The business application layer**

Eventually, there will be no need at all for a network services layer per se, as standards are set (or de facto ones accepted) and as communications architectures become more prevalent. To understand the role of the other two new layers, it is helpful to consider examples of the types of entities and concerns at each.

"Business functions" refers to a set of common facilities for issuing calls to send and receive messages. These calls can be characterized as a request for information and the information in response, each of which is an atomic transaction. Real-time exchanges, such as inventory queries and updates or automated teller machine (ATM) transactions, require the business functions that are driving the initial need for INAs. These business functions include deposits to an account, balance inquiries, updates of spreadsheets, or sending someone a message or a document. They should therefore be able to support a wide variety of sources, such as terminals, programs, microcomputers, and workstations.

Today a bank customer can go to an ATM and say "give me this" or "send this from here to there." It is not similarly possible to simply send a file from any computer to any other, since this calls for a particular logical connection between the two machines. Each such connection has its own protocol, and the number of them grows in proportion to the number of device types to be joined. With an INA, however, the business function "requesting a file" has a local dialog with the D&C layer. Since this layer uses the network services layer, which knows all the required protocols, files can travel from any point in the network to any other. Without that, each business function has to have a separate connection.

Some components of the business application layer are listed below:

- **Atomic transactions.** A business function may consist of several atomic transactions. For example, a person drawing from a checking account with overdraft protection may cause the business function of "debiting" to occur on one computer. The business functions of "overdraft protection" and "crediting" may involve a funds transfer from a reserve credit line on another computer to the processor on which the debiting function initially occurred. The process appears as one transaction to the user. This example can be applied to reservations in the airline industry and to inventory control in manufacturing, retail, and wholesale. As changes to inventories on one computer are made, they affect accounting records on another.
- **Error management.** Failures of business application code must be detected, contained (the atomic transactions insulated from each other), and recovered from by business functions. These functions should take



responsibility for the successful performance of all related atomic transactions. Stand-in business functions must be provided for critical applications.

- **File transfer services.** Such services, in support of the business function, are already among the bread-and-butter services required by data-center applications. Today, the distribution of batch files is a manual operation and may include sending a tape in a cab across town. Instead, an INA should provide automatic distribution. Once a batch job is completed, the batch program should send a transaction to a business function that would cause the results of the batch job to be sent to the other data center. This should occur without operator intervention. While this example concerns large files, microcomputer and workstation users should also be able to submit files for delivery across the network without having to establish a connection to the destination.

- **Software distribution.** Computers will need some means to guarantee that all common software is operating at the same version level. Version maintenance and the distribution and inventory of software can make up one or more business functions.

- **Universal electronic mailboxes.** These locations for users to receive all electronic correspondence will allow users in an IBM DISOSS environment to create a document and send it to another user without having to know whether the addressee resides in the same, or another, DISOSS environment, Wang Office, or DEC All-In-One environment.

- **Library services.** These could contribute to the maturation of electronic mail. Libraries are electronic filing cabinets where users store different types of information objects, such as files, documents, messages, software, and extracts of data. Each object is stored with a set of attributes that may include owner, date, type (such as report or manual), and a descriptive summary. Users can search, display, and select any library item as needed. The location and number of libraries can be organized depending on organizational and business needs. IBM's DISOSS is the best-known library product available. Although DISOSS is capable of storing any type of object, IBM has restricted its use to accept only text documents when accessed from IBM office environments.

Library services should enable businesses to eliminate the need to continuously distribute thousands of pieces of printed literature and to store thousands of individual copies of information, such as updates of policy and procedure manuals.

### **The delivery and control layer**

Services at this layer deal with the following D&C functions: directories, store-and-forward processing, routing, security, statistics, notifications, and logging. The layer determines which business functions are available and can reach them through application interfaces. For example, some business functions may be available only during certain times of day. The D&C function must therefore know where they reside and when they are available. When the business functions are not available, the D&C layer should be able to route

transactions to stand-in business functions on the same, or another, node.

Below are some examples of entities and activities of the D&C layer:

- **Levels of service.** Businesses and users should be able to specify the time of day when delivery should occur and how quickly. They should be able to execute or transmit, for example, microcomputer files (or other objects) or host batch jobs at any given time, without requiring human intervention. D&C functions must provide these capabilities. They should also enable businesses to select the most appropriate level of service to most closely match business operating requirements (for example, for business functions requiring interactive, as opposed to batch, processing).

Alternate network paths must be available to accommodate the different levels of service. The cost of each service may vary relative to the capabilities that a single path can provide, with value-added services being more costly. For example, the cost of a service will vary depending on if a request requires expedited processing and, therefore, a direct link, or if it can be stored at intermediate nodes and forwarded throughout the network until resources become available.

Moreover, the structure of telecommunications tariffs makes it desirable for businesses to be able to schedule information transfers at the time of day when links are cheapest. Intermediate store-and-forward processing is transparent to the user who initiated the information transfer. Of course, businesses should be able to charge back users (divisions, departments, or individuals) for their use of the network. Such a mechanism should also enable corporations to differentiate classes of users and to assign costs to them based on how closely their requested network-service priority levels correspond with business needs.

- **Global network directories.** The information network must maintain a corporate directory of all nodes and their addresses independent of the directory that the underlying communications network uses to establish connections in the first place. Global node and address directories are roughly comparable to post office directories that contain all the zip code information needed to locate a recipient. The global directory must also include the names and addresses of nodes within subnetworks, such as IBM's SNA networks, Wangnets, Decnets, Tandem networks, LANs, and so on.

- **Translation and transformations.** It is necessary that the D&C functions know the internal formats of the objects, such as documents, that may be passed to them by network services. Document structures must be converted into a canonical form prior to their submission to the delivery layer. In this way the business application layer can always work with the same object, regardless of the originating computer. Only business functions dealing with word processing documents need to understand these canonical forms. Another business function, like a checking account inquiry, might need to understand a different set of canonical forms.

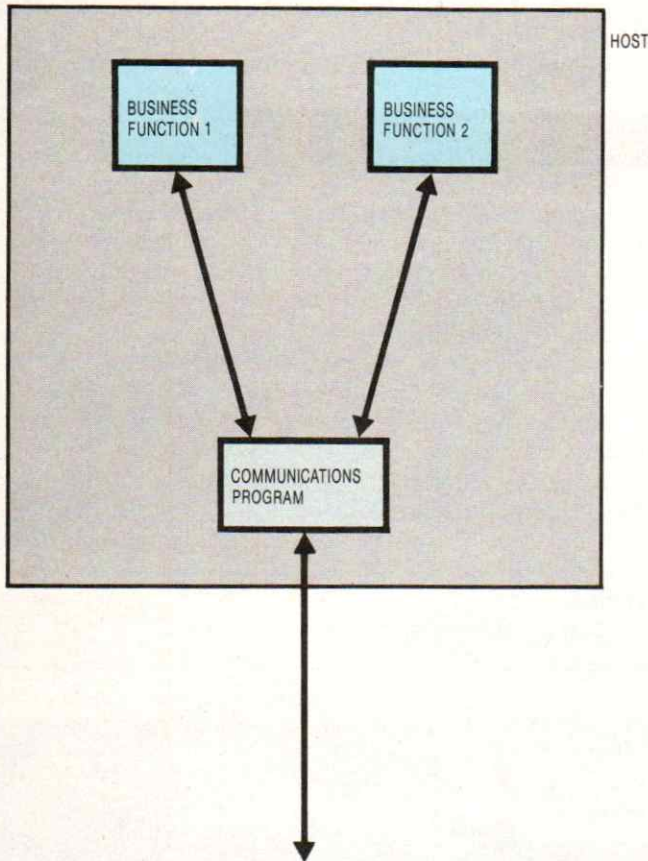
- **Notification procedures.** What is commonly known to users of the postal service as return-receipt-requested



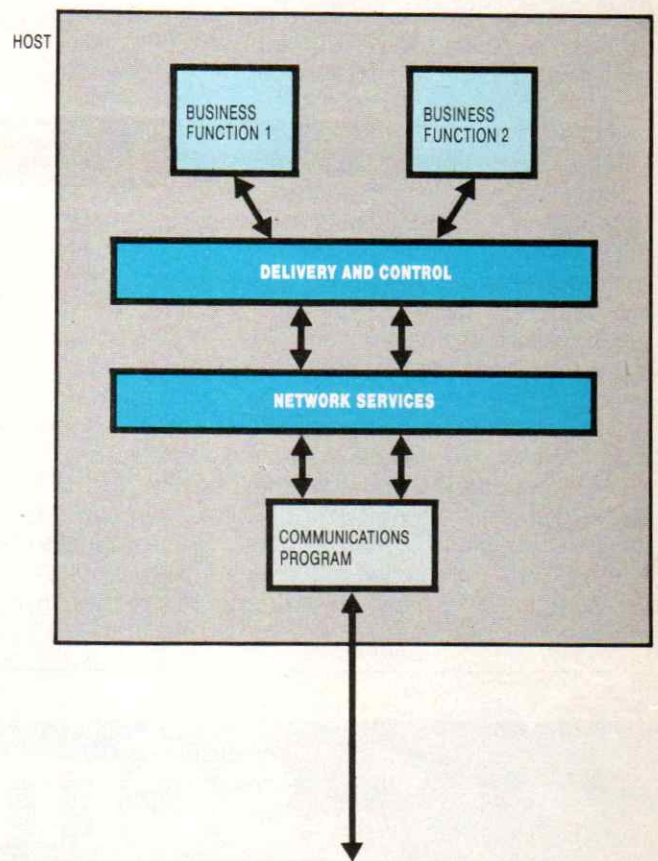
**3. Layers deployed.** Some current host environments are sophisticated enough to separate applications from the supporting communications (A). Even so, one of

these two functions would have to be modified to handle passthrough "postal" messages. Alternatively, separate layers would let this host handle such traffic (B).

(A) WITHOUT AN INFORMATION NETWORK ARCHITECTURE



(B) WITH AN INFORMATION NETWORK ARCHITECTURE



(for example, with certified mail) should be automatically available in a network environment. Both senders and receivers should be automatically notified on completion of such business functions as file or document transfers. Besides automatically notifying the actual recipient, a user should be able to request that additional parties be notified (as in "send this report to Charlie and also notify his boss"). Also, failures that can occur during deliveries and prevent their completion may require notification of different individuals so that the problem can be resolved by the most knowledgeable person. For example, supervisors might need to be informed if certain vital communications of their underlings went awry.

■ **Security.** It is assumed that the required physical security (access to computers and their peripherals) is provided as part of overall premises security. Each individual network within the larger corporate network must be permitted to choose its own local security strategy, such as Resource Access Control Facility (RACF), Access Control Facility (ACF-2), or Top Secret on IBM mainframes. Each network can expect the larger network to respond to these requirements. Security functions must, therefore, validate such things as user identification, data encryption, as well as access

rights with the local security environment.

■ **Recovery management.** This corresponds to recovery management for internal business-code failures. The D&C layer must provide error and recovery management for failures that affect delivery, such as the routing of transactions, files, documents, or messages. Software at the D&C layer must be robust enough to be able to detect and recover from failures without affecting other components (or business functions). In the case of a file or document transfer, users should be allowed to resume their transfers at a checkpoint rather than having to re-establish an entire file or document transfer. The same applies to programs that may handle transactions without operator intervention.

Formatted messages flow from the underlying transport network into the network services layer. This layer transforms these messages into a canonical form that can be processed by the delivery and control layer. Another set of canonical forms should be used between the D&C layer and the business application layer. The network services layer will eventually be rendered unnecessary, so the previous discussion of this layer should suffice.

Recognizing that the next evolutionary step in computer communications is to interconnect or integrate



architectures is one thing. Implementing it is another. However, information network administrators should begin to consider the process, as many of them are reaching an opportune juncture. Most existing code is terminal-to-application oriented and will eventually have to be modified to take advantage of distributed processing, placing an enormous burden on developers. Besides, user organizations will have to rewrite applications both to keep up with evolving user requirements and because some applications will have reached the end of their life cycle.

Depending on the magnitude of the effort needed, user organizations could take advantage of the opportunity to implement the business information architecture discussed in this article. A major benefit of the architecture is that it insulates business and D&C functions from the network services and their underlying communications protocols. This fact provides a migration path to gradually modify existing software.

By moving to this architecture, information network administrators will be able to better insulate their business functions from future changes. The goal of this insulation is to develop new business functions that let users group and regroup dynamically into "circles of interest" at the discretion of the enterprise without having to be in the same place or attached to the same computer.

Moreover, one application program will be able to initiate the execution of another according to the business need. With code at the new layers in each computer, every application program will be able to talk to every other such program (Fig. 3). The only requirement would be an adequate transport network beneath them.

Implementations of the information delivery concept will continue to evolve over the next five to 10 years and will become an important method for integrating diverse applications and diverse computer equipment. Large user organizations will be able to use these implementations to isolate their applications from the underlying network mechanics. This will allow them to choose the most appropriate hardware and software for future applications. ■

*Rudolf Strobl, Ph. D., is a senior consultant at Software Research Corp. (SRC). Strobl is involved in the planning and project management of information delivery products for SRC and Fortune 500 companies. Previously, Strobl was with the Yankee Group, a Boston-based market research and consulting company. Bill Stackhouse is SRC's director of product planning. Prior to joining SRC, Stackhouse was a vice president at the Bank of America. In that position, he was responsible for branch delivery systems.*

# SCITEC

## HIGH SPEED TDM'S

### ACCUNET\* 1.544 MEGABIT

- Install on Microwave, Fibre Optics, and Infrared
- Any Channel from 50 BPS to 256 KPBS
- Menu Formatted Diagnostics, Statistics, Configuration
- Voice & Data
- Teleconferencing
- Time Of Day Module
- High Speed Channel 256 KBPS to 1.344 MEG
- 4 - 128 Channel Expansion
- Alternate Configuration
- Alternate Channel Routing

### DDS-56 KILOBIT

- Any Channel from 50 BPS Unrestricted Mix
- Menu Formatted Diagnostics, Statistics, Configuration
- 4 - 24 Channel Expansion
- Expandable to T-1 Configuration
- RS232/RS423/RS422/V.35 Channel and Trunk Interfaces
- Statistical Muxed Async



## BSPT-1



## BSP56

Scitec Corporation 850 Aquidneck Ave., Middletown, R.I. 02840 (401) 849-4353 TLX AA21730 P.O. Box 30 Newport, R.I. 02840

\* Servicemark of AT & T